VPC - Virtual Private Network

- VPC is virtually isolated private network in the cloud
- VPC is region specific
- As of today max we can create 5 VPCs/Region, however we can increase the limit by writing email to the support team
- VPC is free of cost

Creating VPC

AWS Console → VPC → Your VPCs → Create VPC
Name tag → “WiproVPC”
CIDR → 173.35.0.0/16
And create VPC

Subnet

- Is a smaller network inside VPC.
- Subnet is availability zone specific
- We can create as many subnets as possible
- Subnets are free of cost

Creating Subnet

AWS Console → VPC → Subnets → Create Subnet
Name tag → “BangloreSubnet”
VPC → “WiproVPC”
Availability Zone → No preference
CIDR → 173.35.1.0/24

EC2 (Elastic Compute Cloud)

EC2 is a virtual server in the cloud which has, CPU, RAM, OS and some networking performance.

Use case: My client wants to run website on the cloud

- Create VPC and Subnet

- Get Virtual Server from the cloud
    - 1 CPU
    - 1 GB RAM
    - Linux OS
    - 10 GB HDD
- Install Apache web server on the VM
- Deploy the website code on Apache server

Launch EC2 Instance

AWS Console → EC2 → Launch Instance

1. Choose AMI, AMI is a template containing operating system and few pre installed softwares. Choose (**Amazon Linux**)
2. Choose Instance Type → t2.micro
3. Configure Instance Details
    a. Number of instances → 1
    b. Network → “WiproVPC”
    c. Choose any subnet
    d. Auto asign public ip → enable
4. Add Storage
    a. Storage is nothing but hard disk to the VM
    b. Choose 10 GB
5. Tags
    a. Tags allocates names to the resources and it's also easy to track their bills
    b. Create a tag Key → Name Value → “WiproWebsite”
6. Configure Security Group
    a. It is a virtual firewall to the EC2 instance
        i. It should allow http
        ii. It should allow ssh only from wipro network
        iii. Create a new security group
        iv. Name → “wiprowebsite-sg”
        v. Choose ssh & http source→ anywhere
        vi. Review Instance Launch → Launch
        vii. Select key pair
            1. Create new keypair
            2. Key pair name → hari
            3. Download key pair
            4. Launch Instances

SSH Key Pair

This generates public and private keys, public key is deployed in EC2 and private key is downloaded by us.

For remote server authentication we can use

1. Username with password
2. Username with private key (More secured)

Connecting to remote EC2(VM) instance from our network/laptop

The problem is, our laptop is in different network and EC2 in VPC is in different network and there is no network connectivity between them, to establish network connectivity we have the following options

1. Via internet, using AWS Internet Gateway
2. Configure VPN between our network and VPC (This uses internet)
3. Direct Connect (A dedicated physical cable between our network and AWS)

Accessing EC2 using Internet Gateway

1. Create internet gateway, Internet Gateway is a virtual router between VPC and internet.
    a. AWS Console → VPC → Internet Gateways → Create Internet Gateway
    b. In the Name tag enter → “WiproIGW”
    c. By default the state of Internet Gateway is detached
    d. At a given point of time VPC can be attached with only one Internet Gateway
2. Attach the IGW created in previous step to ***WiproVPC***
    a. Select IGW and click Attach to VPC and select WiproVPC
    b. Attaching IGW to the VPC does not give access for our subnets to internet, to grant internet access to the subnets we need to modify the route table of the subnet.
3. In Order to grant internet access to the subnet, we need to add a route to the internet in its route table.
    a. From VP dashboard, select Route Tables
    b. Under route tables select route table associated with our VPC(WiproVPC).
    c. Select routes
    d. Click Edit
    e. Add another route
        i. In the target → select internet gateway
        ii. In destination 0.0.0.0/0 (Anywhere)

Login to the remote Linux server

- For connecting to linux remote servers we use SSH (Secured Shell)
- For connecting to Windows remote servers we use RDP (Remote Desktop Protocol)

To connect from Windows

- Putty
- Poweshell

- Git bash (Install this)
- Open the git bash
    - ssh -i <path to private key> ec2-user@<public-ip>
    - ssh -i ~/Downloads/hari.pem ec2-user@13.126.3.189

Route Tables

Route table is an virtual router in the cloud, route table contains set of routes. Routers are used in networking to connect different networks to communicate.

- Every subnet in VPC is associated with a "*Route Table*", this enables network communication with internet and communication within VPC with other subnets.
- At the time of creating VPC, a default "Route Table" is implicitly created by AWS, this is also called as Main route table.
- When we create a subnet, this subnet is implicitly associated with Main route table.
- When a new route table is created, it comes with an implicit route, this implicit route enables the subnets associated with this route table to communicate with other subnets in the same VPC.
- We can associate multiple subnets to the same route table.
- A subnet can be associated with only one route table.
- We also can create custom route tables
- We cannot delete main route table
- We can make other route table as Main route table

Public IP And Elastic IP

Public IP/ Elastic IP are used to expose the server to internet.

Public IP

Public Ip is dynamic public, If we stop the server the public IP is released back to public IP pool, when we start the server AWS picks a new public IP from the pool.

Elastic IP

Elastic IP is a static ip, even if we stop the server IP remains with us.
We can disassociate EIP from the instance and can associate with other instance.

Note: *If EIP is associated **with non running EC2 instances** then we incur minimal charges.*

Allocating EIP and associating with EC2

VPC → Elastic Ips → Allocate new address
Select EIP → Actions → Associate Address → Select EC2 Instance

Public And Private Subnets

Public Subnet

A subnet which is directly exposed to internet is called as public subnet. We make subnets public by adding Internet Gateway.
We use public subnets for internet facing servers like web servers.

Private Subnet

A subnet which is not directly exposed to internet is called as private subnet.
For example, database servers, build servers, any server which should not be exposed to internet should be part of private subnet.

Default VPC

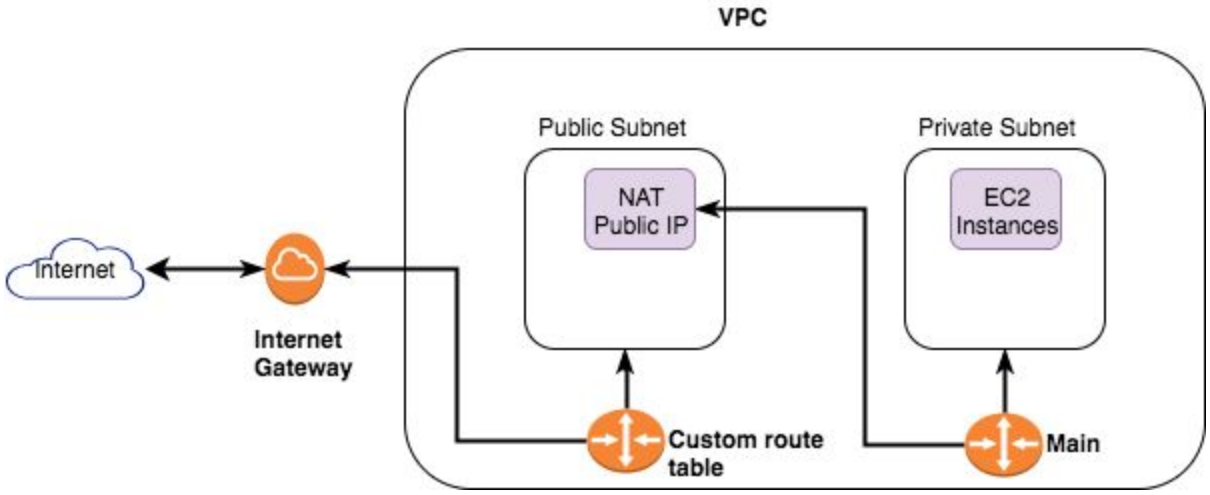
- Default VPC is ready to use VPC
- Default VPC is present in all the AWS regions
- For each availability zone has an subnet
- All subnets in default VPC are public
- Public IP is auto enables for all the subnets
- We can delete default VPC, however it's not recommended
- If we Accidentally deleted default VPC we cannot recreate on our own, we have to create a support ticket to recreate it.

How to give internet access to private subnet

- We can use either **NAT Instance** or **Nat Gateway,**
- This provides only outbound access and no inbound access
- Internet Gateway give both inbound and outbound access
- NAT Instance and Nat Gateway are more secure than Internet Gateways.
- In case of IGW every machine requires public ip which is a limited resource over internet
- In case of NAT only one public IP is required

NAT Instance

NAT stands for **N**etwork **A**ddress **T**ranslation, NAt instance is EC2 instance configured with NAT settings, it is used to grant internet access to the private subnets, It allows only outbound traffic and restricts inbound traffic.



- NAT must be launched in public subnet with public IP
- Modify route table of private subnet and add route to NAT instance
- Internet traffic flow for private subnet
    1. An internet traffic is initiated by EC2 in private subnet.
    2. This traffic is routed to NAT instance present in public subnet
    3. NAT forwards the traffic to internet via Internet Gateway
    4. Internet responds back to NAT instance
    5. NAT forwards the response back to EC2
- Disable **Source/destination** check for NAT instance
- In AWS all EC2 instance are enabled with source/destination check i.e. EC2 must be the source and destination for the traffic otherwise the traffic is blocked.
- In case of NAT it is not the source/destination of the traffic, so we need to disable source/destination check.

NAT Implementation

- Create VPC → NATDemoVPC
- Create PublicSubnet (With Custom route table)
- Create PrivateSubnet (With Main route table)
- Launch NAT instance in public subnet
    1. EC2 → Launch Instance
    2. Select **Community AMIs**
    3. Search with nat and select one from the result
    4. Select t2.micro

5. Select NATDemoVPC
6. Select Public Subnet
7. Enable public IP
8. Under tags Key → Name and Value → Nat Instance
9. For security group allow all traffic from anywhere
10. Review and launch
11. Select key pair and launch
12. Disable source and destination check for NAT.
    a. Select Nat Instance
    b. Actions → Networking → Change Source/Dest. Check
    c. And click disable

- Select route table of private subnet
- Add the route to the Nat Instance
  - Destination → 0.0.0.0/0
  - Target → Nat Instance
  - Save

Testing The configuration we did

- Launch EC2 in private subnet
- ssh into this EC2
- ping google.com

Difference between NAT Instance & NAT Gateway

NAT Instance:

- It is not Highly available, For making NAT Instance HA, we have to maintain multiple NAT instances and write a script which detects the failure and automatically failovers to the standby NAT instance.
- We have to maintain the NAT instance
- If we face performance issues we need to replace small instance type with big instance type.

NAT Gateway:

- Highly Available, If primary component fails it auto failovers to the standby component.
- Highly scalable, it automatically resizes based on the demand.
- It's completely managed by AWS

Note: NAT is not available under free tier (Charges 2000/Month approx)

Configuring new Users in EC2

1. Launch EC2 instance
2. Ask the new user(for ex: krishna) to generate private and public keys on his laptop
    a. Open ssh tools (Git Bash, Putty, etc)
    b. ssh-keygen -f krishna
3. Create new user in ec2 instance with name krishna
    a. sudo useradd krishna
4. Configure public key of the krishna under his user in EC2
    a. su krishna
    b. cd /home/krishna/
    c. mkdir .ssh
    d. `chmod 700 .ssh`
    e. cd .ssh
    f. Open authorized_keys and paste public key of krishna in this file

VPC Wizards

Wizards simplifies our VPC, IGW and NAT configurations.
Currently we have 4 types of wizards

1. VPC with public subnet
2. VPC with public and private subnet
3. VPC with Public and Private Subnets and Hardware VPN Access
4. VPC with a Private Subnet Only and Hardware VPN Access

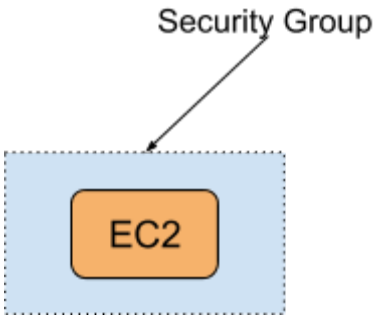
Securing VPC

We secure VPCs in two ways

1. Security Groups
2. Network ACLs (Access Control List)

Security Groups and its Characteristics

Security group is a virtual firewall to our EC2 instance, using security groups we secure EC2 instances. Every security group has number of rules.



Security Group Characteristics

1. Every instance must have at least one security group
2. Same security group can be associated with multiple EC2 instances
3. Every EC2 can have max 5 security groups/eni
4. Security groups has both inbound and outbound rules
5. *Security groups are stateful*
    a. If a traffic is initiated from internet, this traffic is validated by inbound rules of security groups, no matter what is the outbound rule the traffics leaves out.
    b. If a traffic is initiated from EC2, this traffic is validated by outbound rules of security groups, no matter what is the inbound rules the traffics comes in.
6. When we update a rule, it takes effect almost immediately(1-2 seconds delay is expected)
7. Security groups does not have explicit Allow/Deny, Rules we add is to allow and others are implicitly denied.
8. Security group Source
    a. My IP → This traffic is allowed only from My IP
    b. Custom
        i. CIDIR (For example company network)
        ii. We also can use a specific IP
        iii. We also can use security group in the custom field, i.e. all EC2 instances with this security group can access this instance

Network ACLs and its Characteristics

1. Network ACL is firewall acts at subnet level
2. When VPC is created NACL is implicitly created, this is called as default NACL
3. ***Default NACL allows all inbound and outbound traffic***
4. Any subnet created is implicitly associated with default NACL
5. Multiple subnets can be associated with one NACL
6. One subnet can be associated with only one NACL
7. Every rule can be explicitly allowed or denied

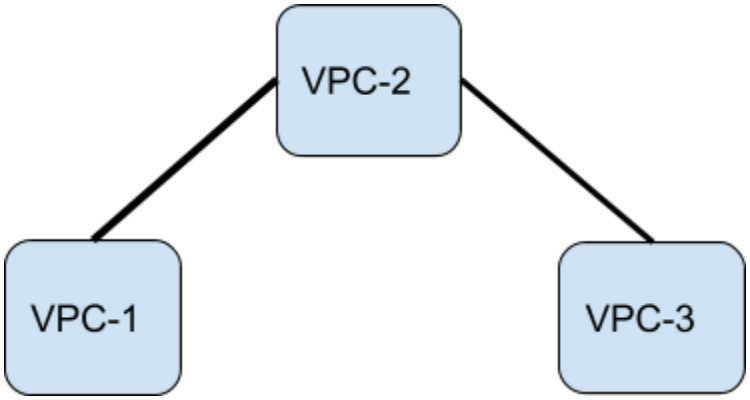
8. ***Network ACLs are stateless,*** i.e. inbound traffic is controlled by inbound and outbound traffic is controlled by outbound rule.
9. We have options to block an ip address/network using NACL

How Network ACL rules are evaluated?

- Rules are evaluated in ascending order
- When matching rule is found rule evaluation stops by applying (Allow/Deny)

VPC Peering

- Using this we establish network connectivity between VPCs
- VPC peering can be done within same or different accounts
- Both VPCs must be in same region
- CIDR notations of the VPCs should not collide
- For connecting VPCs in different regions we can either use VPN or Direct connect.
- Transitive peering is not supported



For example we have peering connection between VPC-1 → VPC-2 and VPC-2 → VPC-3, transitive peering is not supported, i.e VPC-1 cannot communicate with VPC-3

VPC Peering Demo

1. Create two VPCs
2. At least one subnet in each VPC
3. VPC → Peering Connections → Create Peering Connection
4. Name → Peering Demo
5. Select VPCs
6. Create
7. Select peering connection → Actions → Accept
8. Modify routing tables in both the VPC

EC2 Userdata

Using this option we can run scripts at EC2 launch time. There are many use cases for this, for example we wanna configure our servers with chef/puppet we need chef/puppet agents on our machines this can be achieved using userdata.

Example: Using user data

1. Install apache server
2. Start and enable apache server
3. Deploy a sample html file on the apache server

Launch EC2 and at step 3 under user data paste this script

```
#! /bin/bash

yum install httpd -y
service httpd start
chkconfig httpd on
echo "<h1> User data example </h1>" > /var/www/html/index.html
```

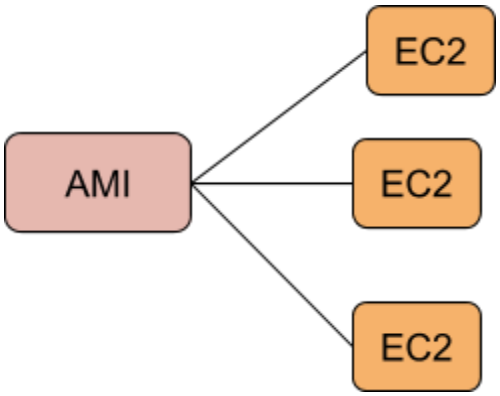
Note: ***Do not explicitly mention sudo, all the scripts in user data runs internally using sudo***.

EC2 Termination protection

This option disables accidental termination of EC2 instances.
This option can be enabled while launching EC2 or on an existing EC2.
Select EC2 instance → Actions → Change termination protection → Enable/Disable

Custom AMI (Amazon Machine Image)

AMI is template which contains, Operating System and pre installed softwares



We want a custom AMI which contains

- Apache Installed, Started, Enabled
- Our Application installed no Apache server
- By default AMI created by us is private
- We also can share with other AWS accounts (We can find under AMIs)
- We also can make it public (We can find under **Community AMIs)**
- When we share this image, this image is accessible from the same region.

Creating Custom AMI

- Launch EC2 and install what all you need in the image
- From above EC2 create an image

Launching EC2 instances from the AMI

EC2 Instance resizing

Instance resizing is a way to scale up or scale down our EC2 instances.
Note: *We must stop the instance before resizing*.
Select Instance → Actions → Instance Settings → Change Instance Type

EBS Volumes

EBS stands for Elastic Block Store, EBS is Highly Available and highly durable block level storage. By default AWS maintains data across 2 different physical components.
Block level storage is used for File systems, and we also can install and run software applications.
EBS characteristics

1. EBS is persistent store, i.e. permanent storage.
2. To attach a volume both volume and EC2 must be in the same AZ
3. We can add multiple volumes to the instance.
4. We can migrate a volume in one AZ to another AZ by taking a snapshot
5. We can detach a volume from a EC2 and attach to other instance
6. We can take snapshots (backups), and we also can restore from snapshot.
7. Snapshots are stored in S3 (Simple Storage Service)

8. A volume cannot be attached to more than one instance at a time.
9. Volumes support incremental snapshots, i.e. whatever is modified only that is backed up
10. We can resize volumes, but we cannot downsize.
11. EBS works only as a sub service of EC2
12. We also can encrypt EBS volumes

Creating EBS Volume

We can create volumes in multiple ways

- At the time of launching ec2 instance.
- We also can create a volume from volumes from the EC2 dashboard
- We can create volumes from snapshots

Resizing EBS volume

EC2 → Volumes → Select Volume → Modify → changes the values and save
Note: We can increase the size, but we cannot decrease it.

Adding new volume and mounting it to EC2

- Create a new volume
- Attach the volume to EC2
- Login to EC2
    - lsblk (this command displays all the volumes)
    - Create a file system
        sudo mkfs -t ext4 /dev/xvdf

    Note: the command to check if filesystem is already created
    `sudo file -s /dev/xvdf`
    - Create a mount point
        sudo mkdir /xyz
    - Mount new volume
        sudo mount /dev/xvdf /xyz
    - Make the above changes persistent
      Open sudo vi /etc/fstab
      Add the following entry to the file and save
      /dev/xvdf /javahome ext4 defaults 0 0

Taking EBS snapshots

EC2 → Volumes → Select the volume → Actions → Create snapshot

Restoring Volumes from snapshots

EC2 → Snapshots → Select the snapshot → Actions → Create volume

While restoring we can choose

- Different volume type
- Different availability zone
- Different size

EBS Volume Types

1. General Purpose SSD (Solid State Disk)
2. Provisioned IOPS SSD
3. Throughput optimized HDD
4. Cold HDD
5. Magnetic

General Purpose SSD (Solid State Disk)

These volume type supports large set of workloads.
Performance:

- IOPS (Input Output Operations/Second)
- Minimum 100 and max 10000 IPOS
- This volume types can perform above baseline when required, by spending credit points, EBS volumes accumulates credit points when it is idle.
- This volume type is good fit for (Dev, Testing, Staging environments)
- Oracle, Sybase, DB2, MySQL, mongodb etc..
- We also can use it for production

Provisioned IOPS

This volume type provides extreme performance, it designed for machine critical applications.

- Max IOPS, 20000/volume
- We can get upto 65000 IOPS by attaching multiple volumes
- By configuring RAID we can increase IOPS beyond 65000
- Oracle, Sybase, DB2, MySQL, mongodb etc..
- IOPS is the performance criteria not the throughput
- Random reads and writes
- Good choice for production workloads, not recommended for lower environments(dev, testing, staging) in order to save the cost.
- IO size is small

Throughput Optimised HDD

- This volume type is a good fit for processing big data.
- Like Data warehousing, Big Data, Informatica etc…
- Throughput and sequential reads are dominant performances attributes
- Gives faster access to data

- Use this type where we access data more frequently

Cold HDD

- Works similar to Throughput Optimized HDD, storage cost is cheaper
- But this type has additional retrieval charges
- Good option for infrequently accessed data
- Gives faster access to data

Magnetic

Magnetic is an old generation storage type, we should always consider other types except if our application is specially optimized for this type.

Instance Store

Is a temporary store, data stored on this store is lost when we poweroff the Instance
Storage cost is very very cheap compared with EBS
Use this types to store temporary data.

Launching EC2 with instance store

EC2 → Launch Instance → Community AMIs → filter by root device type(Instance store)

Instance Types

EC2 Instances (Purchase Options)

- On-Demand Instances
- Reserved Instances
- Spot requests
- Dedicated Hosts
- Scheduled Instances

On-Demand Instances

- These instances are charged per hour basis
- Even if we use this server for one minute, it is counted as one hour.
- When instance is not running, there are no charges
- Even if instance is stopped, bill counts for EBS volumes attached to the instance.
- Use this instance type when we need instances for short duration (1 day, 1 week, one month)

Reserved Instances

We can reserve instances either one or three years, the advantage is we can save upto 70% of the cost when compared with On-Demand instances.
One reserved purchase is done, there is no option to terminate the contract. However we have option to sell it over AWS marketplace.
Payment options

- All upfront
- Partial upfront
- No upfront

Spot requests

Amazon EC2 Spot instances allow you to bid on spare Amazon EC2 computing capacity for up to 90% off the On-Demand price

Dedicated Hosts

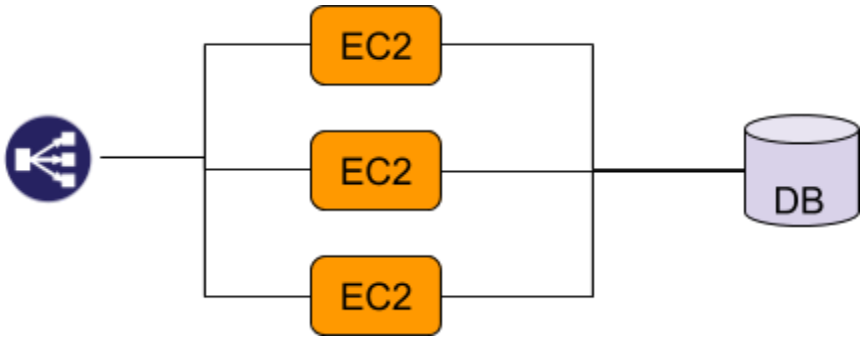
A Dedicated Host is a physical EC2 server dedicated for your use. Dedicated Hosts can help you reduce costs by allowing you to use your existing server-bound software licenses, including Windows Server, SQL Server, and SUSE Linux Enterprise Server (subject to your license terms), and can also help you meet compliance requirements

Scheduled Instances

We can buy instances based on our schedule, i.e. we need servers daily, weekly, monthly only in the dedicated hours for example 2-3 hours/day, then this type is the good fit.

Elastic Load Balancers (ELB)

Load balancers are used to Improve application performance and we also can make our application high availability.



ELB Characteristics

- ELB is highly available and highly scalable load balancing service from AWS
- AWS maintains redundant copies of ELB, if any AZ fails it automatically failovers to the ELB in different AZ.
- ELB is region specific, i.e. it can load balance instance present in same region.
- ELB supports both external (Internet facing) and internal(not available over internet)
- ELB does health checks on instances and routes traffic to healthy instances, if it finds any instance unhealthy that instance is taken out of rotation, if ELB finds unhealthy instances are healthy, it brings them automatically into rotation.
- ELB, Supports SSL termination.
- ELB can be secured using Security Groups
- ELB types
    - Classic Load Balancer (Old Generation Load Balancer)
    - Application Load Balancer(Specially designed for microservices/ Docker)
    - Network Load Balancer

Configuring ELB (Classic Load Balancer)

For this demo we need two EC2 instances in two AZs.
This instance must have Apache + A sample html file deployed on apache

Scaling our application

We have two types of scaling

- Vertical scaling(Increasing instance size)
- Horizontal scaling (Adding additional servers to the environment)

Auto Scaling

Auto Scaling makes sure our environment has desired capacity of EC2 instances when it is needed.

Auto scaling supports only horizontal scaling.
If auto scaling finds unhealthy instances it is replaced with new instances by terminating the old instance.
Note: Auto scaling service is free, however instances launched by auto scaling is chargeable.

Terminology

Auto scaling Group:

It is a logical group of EC2 instances which participates in the auto scaling.
This group contains
Minimum - Minimum number of instances this group must contain always
Maximum - Maximum number of instance that can participate in auto scaling.
Desired: Is decided at runtime, based on the scaling policies

Note: We can use auto scaling with fixed group size

Launch Configuration:

Is a templated used by auto scaling for launching new instances.

Launch configuration contains

1. AMI, This has to be our custom image, which contains
    a. App dependencies + Web server + Latest application code
2. Instance Type
3. EBS volume
4. Security Group
5. IAM role
6. ssh keypair

We cannot update launch configuration, we only can configure a new one and attach to ASG

Creating Auto Scaling Group

1. Configure Launch configuration with above details
2. Group Name → demo-auto-scaling-group
3. Group size → 2
4. Select VPC and subnets
5. Advanced Details
    a. Select ELB
    b. Health check type is ELB
    c. Select use scaling policies to adjust the capacity of this group

Identity and Access Management - IAM

Using IAM

- Manage users
- Manage groups
- Manage permissions (user level & Service level)

- Managing Roles
- We can federate with, AD, LDAP, OID or any other SAML implementations
- We also can integrate with identity providers like (Amazon, Facebook, Google, LinkedIn, etc)
- We also can grant cross-account access
- We can enable MFA (Multi Factor Authentication)

Examples:

Create IAM user and grant admin permissions

IAM Dashboard → Users → Add User
Select AWS access type → AWS Management Console access
Select custom password
Select, Require password reset
Attach existing policies directly

Create Groups

Groups → Create New Group → Select a policy and create.

Adding Users to the Group

Create User → Select the group and add it

MFA Multi Factor Authentication

This provides additional level of security to our account, it is always recommended to enable this for all the users.
We can link two types of tokens

1. Hard token (RSA device)
2. Soft token
3. The following apps are supported as a soft token.
    a. Android, iphone, blackberry supports
        i. Google Authenticator
    b. Windows phone supports
        i. Authenticator

Activating MFA (Multi Factor Authentication)

First we need the install the app on the mobile
Open the app and click BEGIN SETUP
From AWS IAM Dashboard

Select → Activate MFA on your root account → Manage MFA → A virtual MFA device → Next → Next → From mobile app scan the QR codes and enter the two numbers one after another, number displayed on the device.

AWS IAM Policies

Policy is a JSON document, which talks about the permissions

Policy Syntax

```
{
  "Statement":[{
    "Effect":"effect",
    "Action":"action",
    "Resource":"arn",
    "Condition":{
      "condition":{
        "key":"value"
        }
      }
    }
  ]
}
```

There are various elements that make up a statement:

- Effect: The effect can be Allow or Deny. By default, IAM users don't have permission to use resources and API actions, so all requests are denied. An explicit allow overrides the default. An explicit deny overrides any allows.
- Action: The action is the specific API action for which you are granting or denying permission. To learn about specifying action, see Actions for Amazon EC2.
- Resource: The resource that's affected by the action. Some Amazon EC2 API actions allow you to include specific resources in your policy that can be created or modified by the action. To specify a resource in the statement, you need to use its Amazon Resource Name (ARN). For more information about specifying the ARN value, see Amazon Resource Names for Amazon EC2. For more information about which API actions support which ARNs, see Supported Resource-Level Permissions for Amazon EC2 API Actions. If the API action does not support ARNs, use the * wildcard to specify that all resources can be affected by the action.
- Condition: Conditions are optional. They can be used to control when your policy is in effect. For more information about specifying conditions for Amazon EC2, see Condition Keys for Amazon EC2.

To learn policies we need to know JSON
JSON stands for *Java Script Object Notation,*
JSON is used for couple of reasons

1. To exchange information between applications (Facebook server and facebook mobile app)
   All mobile apps exchange data with their server using JSON
2. To represent/store data

JSON syntaxes

```
{
  "name": "John",
  "age": 31,
  "city": "New York"
 "isManager": true
}
```

The above is an JSON document, which represents three properties name, age and city.

JSON Data types

- Text
- Number (Integer and floating point)
- Boolean (true/false)
- List
- Objects and Nested Objects

```
{
 "name": "John",
 "age": 31,
 "salary": 10000.5,
 "isManager": false,
 "address": {
  "houseNo": "12-45",
  "landmark": "Dentalcollegeroad",
  "state": "KA",
  "pin": 560037
 },
 "languages": [
  "English",
  "Hindi",
  "French"
 ]"
}
```

ARN (Amazon Resource Names)

```
arn:aws:ec2:ap-southeast-2:aws-account:resource-path
Few examples, the following resource represents ec2 with instance id

arn:aws:ec2:ap-southeast-2::instance/i-05c217fc44889449e
```

Requirement:
Create a policy document which allows, start and stop permissions for a specific instance

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "Stmt1506566739000",
         "Effect": "Allow",
         "Action": [
            "ec2:StartInstances",
            "ec2:StopInstances"
         ],
         "Resource": [
            "arn:aws:ec2:us-west-2:425139712349:instance/i-0499de4a913a8cf75"
         ]
      },
      {
         "Sid": "describeinstances",
         "Effect": "Allow",
         "Action": [
            "ec2:Describe*"
         ],
         "Resource": [
            "*"
         ]
      }
   ]
}
```

Requirement:
Create a policy document which allows, start and stop permissions for a specific instance type, i.e. t2.micro

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "Stmt1506912387000",
         "Effect": "Allow",
         "Action": [
            "ec2:DescribeInstances"
```

```
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Sid": "Stmt1506912495000",
            "Effect": "Allow",
            "Action": [
                "ec2:StartInstances",
                "ec2:StopInstances"
            ],
            "Condition": {
                "StringEquals": {
                    "ec2:InstanceType": "t2.micro"
                }
            },
            "Resource": [
                "*"
            ]
        }
    ]
}
```

AWS CLI (Command Line Interface)

We also can manage AWS resources from command prompt, CLI is import because we use it for automation.

Installing and configuring AWS CLI

1. Install AWS CLI
2. Create a user with programmatic access in IAM
    a. access key ID (Username)
    b. secret access key(Password)
3. Configure AWS CLI with above credentials, by running following commands
    a. aws configure
        This prompts for access key ID and secret access key and region

Note: From this machine we can, we can access services in AWS via CLI, API, SDK (programs) Which internally uses the credentials configured via CLI.

AWS CLI Example

1. Launch ec2 from AWS CLI

aws ec2 run-instances --image-id ami-aa5ebdd2 --instance-type t2.micro

2. Describe all the instances

aws ec2 describe-instances

3. Stop EC2 instance

aws ec2 stop-instances --instance-ids i-0fda0100e2a447421

4. Start EC2 instance

aws ec2 start-instances --instance-ids i-0fda0100e2a447421 i-0bde64e68ef435562

5. Create EBS volume

create-volume --availability-zone us-west-2a --size 10 --volume-type gp2

6. Attaching a volume to instance

aws ec2 attach-volume --instance-id i-01b71ab575ae3f5e5 --volume-id vol-09f368f82e0ced451 --device xvdh

IAM Roles

Role works similar to users, but we assign role to any entity.

Create a Role

Roles → Createrole → Select EC2 → EC2 → Next → Select policy → Review → create policy

Use cases for roles

1. If one AWS service wants to access other aws service it needs permissions, that is assigned by creating a role.
   Example: EC2 instance wants to access S3, EC2 needs permissions, it is grated by creating and attaching a role to EC2.
2. IAM roles is also used for cross account access

IAM Identity federation

We can federate with the users who are not present in AWS, i.e. for example a company is maintaining users and their credential in AD (Active Directory), we can grant access to users in AD to AWS, this is done with identity federation in AWS.

IAM integration with Identity providers

We can integrate AWS with identity providers like (Facebook, google, linkedin, Amazon, etc)

AWS CloudWatch

Cloud watch is monitoring tool, we can monitor resources and applications, we also can schedule tasks using cloud watch, we also can gather logs generated by applications, we also can setup alerts.

Monitoring:

We can monitor aws resources like (EC2, EBS, ELB, S3, RDS, etc).
Cloudwatch automatically collects lots of metrics, for EC2 (CPU Utilization, Disk Reads, Disk Writes, Network packets in, network packets out, etc..)
We also can publish custom metrics and publish to the cloud watch
Note: *Cloud watch does not collect Memory and Volume metrics*
We also can create custom metrics and publish.
AWS supports two types of monitoring

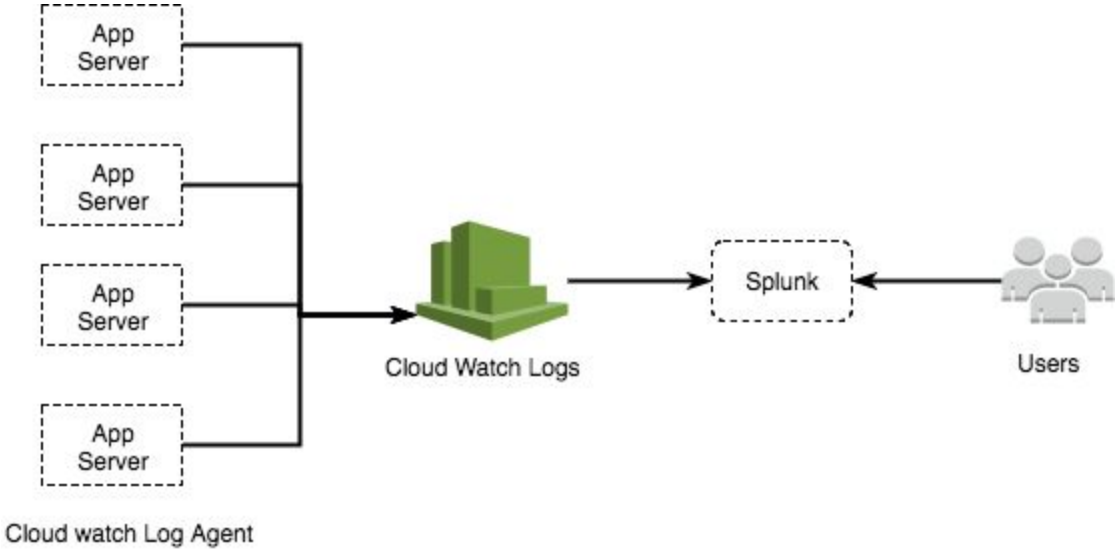
- Basic monitoring, free (metric values are published to cloudwatch every 5 minutes)
- Detailed monitoring, is charged (metric values are published to cloudwatch every minute)

Alarms:

Based on metric values we can create alarms and send notifications.
Alarms can be integrated with

- Notifications(Email/SMS)
- Auto Scaling Groups
- EC2 Actions

Cloudwatch Logs:



Cloud watch, Scheduling Tasks

AWS Lambda (AWS Development)

AWS lambda Is ***serverless architecture***, i.e. we have to write the code and submit it to AWS, AWS takes care of provisioning servers.
As of today it supports four languages

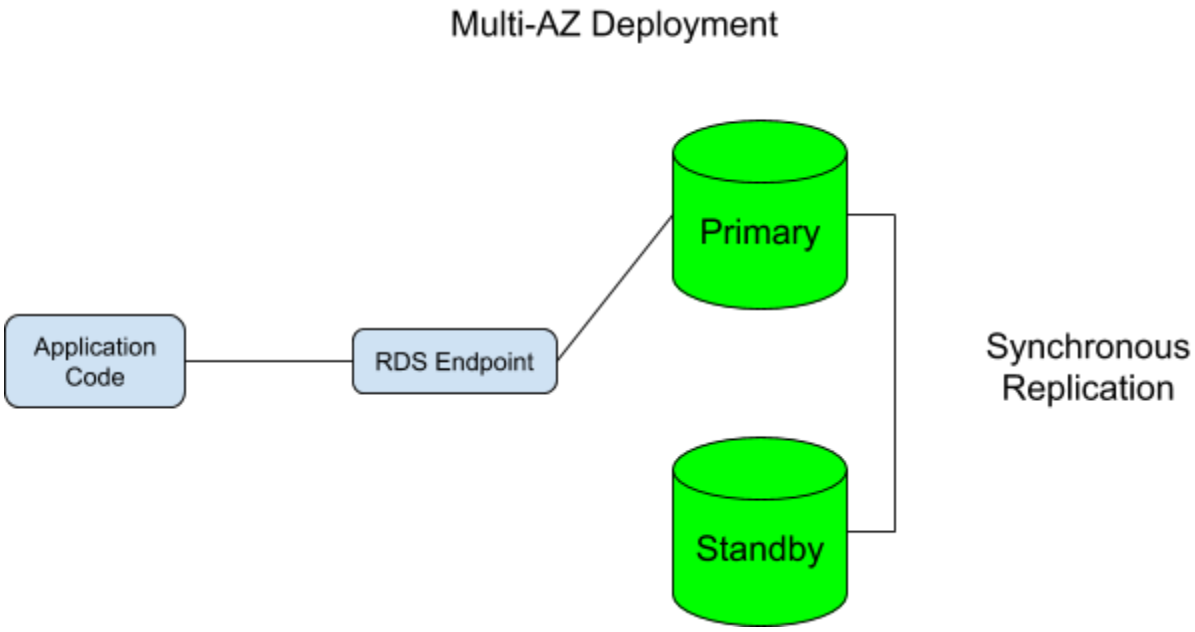
- Node JS
- Python (Most prefered language)
- Java
- C#

Databases

RDS (Relational Database Service)

Amazon Relational Database Service (Amazon RDS) makes it easy to set up, operate, and scale a relational database in the cloud. It provides cost-efficient and resizable capacity while automating time-consuming administration tasks such as hardware provisioning, database setup, patching and backups. It frees you to focus on your applications so you can give them the fast performance, high availability, security and compatibility they need.

RDS Multi-AZ Deployments

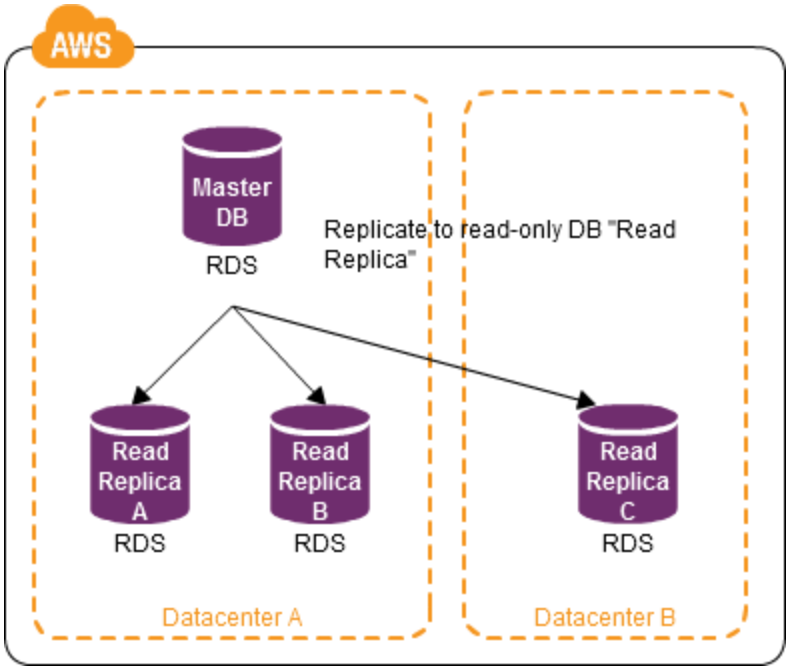


Amazon RDS Multi-AZ deployments provide enhanced availability and durability for Database (DB) Instances, making them a natural fit for production database workloads. When you provision a Multi-AZ DB Instance, Amazon RDS automatically creates a primary DB Instance and synchronously replicates the data to a standby instance in a different Availability Zone (AZ)

Each AZ runs on its own physically distinct, independent infrastructure, and is engineered to be highly reliable. In case of an infrastructure failure, Amazon RDS performs an automatic failover to the standby (or to a read replica in the case of Amazon Aurora), so that you can resume database operations as soon as the failover is complete. Since the endpoint for your DB Instance remains the same after a failover, your application can resume database operation without the need for manual administrative intervention.

Note: Multi-AZ deployments provide enhanced availability and durability, it's not meant for improving application performance.

Amazon Read Replicas



Amazon RDS Read Replicas provide enhanced performance and durability for database (DB) instances. This replication feature makes it easy to elastically scale out beyond the capacity constraints of a single DB Instance for **read-heavy database workloads**. You can create one or more replicas of a given source DB Instance and serve high-volume application read traffic from multiple copies of your data, thereby increasing aggregate read throughput. Read replicas can also be promoted when needed to become standalone DB instances.

Working With Backups

Amazon RDS creates and saves automated backups of your DB instance to S3(Simple Storage Service). backing up the entire DB instance and not just individual databases. Default retention period is 7 days & it can be max upto 35 days, if you do not need automated backup then retention period should be 0.

Amazon RDS creates automated backups of your DB instance during the backup window of your DB instance. If necessary, you can recover your database to any point in time during the backup retention period.

You can also backup your DB instance manually, by manually creating a DB snapshot

The Backup Window

Automated backups occur daily during the preferred backup window. If the backup requires more time than allotted to the backup window, the backup continues after the window ends, until it finishes.

During the automatic backup window, storage I/O might be suspended briefly while the backup process initializes (typically under a few seconds). You may experience elevated latencies for a few minutes during backups for Multi-AZ deployments.
For MariaDB, MySQL, Oracle, and PostgreSQL, I/O activity is not suspended on your primary during backup for Multi-AZ deployments, because the backup is taken from the standby.
For SQL Server, I/O activity is suspended briefly during backup for Multi-AZ deployments.
If you don't specify a preferred backup window when you create the DB instance, Amazon RDS assigns a default 30-minute backup window which is selected at random from an 8-hour block of time per region.

Restore From a DB Backups

If any failure occurs to the database instance, we can restore it from the snapshots.

You can create a DB instance by restoring from this DB snapshot. When you restore the DB instance, you provide the name of the DB snapshot to restore from, and then provide a name for the new DB instance that is created from the restore. You cannot restore from a DB snapshot to an existing DB instance; a new DB instance is created when you restore.

Redshift

Amazon Redshift is a fast, fully managed data warehouse that makes it simple and cost-effective to analyze all your data using standard SQL and your existing Business Intelligence (BI) tools. It allows you to run complex analytic queries against petabytes of structured data, using sophisticated query optimization, columnar storage on high-performance local disks, and massively parallel query execution. Most results come back in seconds. With Amazon Redshift, you can start small for just $0.25 per

hour with no commitments and scale out to petabytes of data for $1,000 per terabyte per year, less than a tenth the cost of traditional solutions.

Elasticache

Amazon ElastiCache is a web service that makes it easy to deploy, operate, and scale an in-memory data store or cache in the cloud. The service improves the performance of web applications by allowing you to retrieve information from fast, managed, in-memory data stores, instead of relying entirely on slower disk-based databases. Amazon ElastiCache supports two open-source in-memory engines:

- Redis - a fast, open source, in-memory data store and cache. Amazon ElastiCache for Redis is a Redis-compatible in-memory service that delivers the ease-of-use and power of Redis along with the availability, reliability and performance suitable for the most demanding applications. Both single-node and up to 15-shard clusters are available, enabling scalability to up to 3.55 TiB of in-memory data. ElastiCache for Redis is fully managed, scalable, and secure - making it an ideal candidate to power high-performance use cases such as Web, Mobile Apps, Gaming, Ad-Tech, and IoT.
- Memcached - a widely adopted memory object caching system. ElastiCache is protocol compliant with Memcached, so popular tools that you use today with existing Memcached environments will work seamlessly with the service.

S3 - Versioning

1. To avoid accidental deletion of objects
2. It maintains multiple versions of same objects
3. We have control to retrieve/delete any version.
4. When we delete an object it adds delete marker, instead of actually deleting an object.
5. Downside is, every version occupies its complete storage capacity

6. Ones versioning is enabled there is no option to disable, we can only suspend new versions.

S3 - Lifecycle management

You can manage an object's lifecycle by using a lifecycle rule, which defines how Amazon S3 manages objects during their lifetime.
Using lifecycle rules we can autom transition of objects to cheaper storage like (Standard IA, Glacier) to save money.
Using lifecycle rules we can autom expiration of objects.

Lifecycle use cases:
Requirement -1

I want to expire previous versions, if it is 30 days old

Requirement -2
We are working for a bank and they are storing log files in S3, to same cost, they want the following lifecycle
Standard --after 6 months → IA --- after 3 months → Glacier

S3 - logging

This allows us to track all API activities happening on this bucket

S3 - Hosting Static Websites

We can host static websites without need of servers.
It does not support dynamic websites like servlets, jsp, python django, .Net etc...

S3 - Events

Receive notifications when specific events occur in your bucket
Usecases:

1. Resize image when image is uploaded to the S3 bucket.
2. Create thumbnails for the images
3. When a CSV file is uploaded to S3, read the file content and store in dynamodb.
4. Whenever a mp4 videos are uploaded send email notification

Cross Region Replication

IF we upload an object into source bucket, it can be automatically replicated into destination bucket which is in different region.
Usecases:

1. Disaster recovery
2. Improve application performance
3. If there is a government rule which mandates the data should be available in a s specific geographical regions

Note:

1. Cross Region Replication works only if versioning is enabled on both the buckets.
2. It replicated only future uploads, it will not replicate the objects which exists before enabling CRR